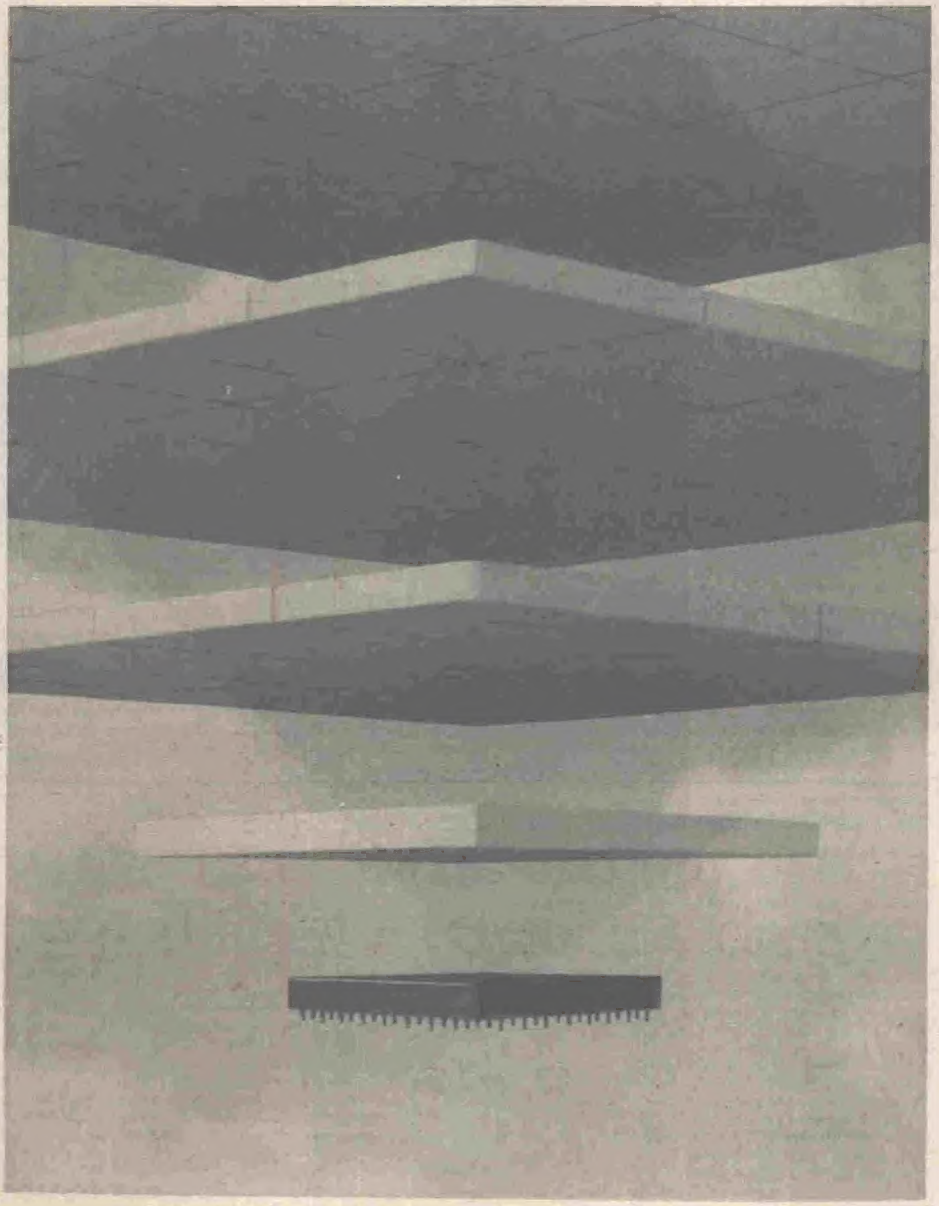
ก้าวต่อไปกับ 80486

แสดงให้เห็นถึง สถาปัตยกรรมของ ซีพียู 80486 และ ข้อแตกต่างกับ 80386



เราเคยเรียนรู้และเริ่มต้นกับ 80386 ไปบ้างแล้ว มาคราวนี้จะนำท่านผู้อ่านก้าวไปกับเทคโนโลยี 80486 กันเลยดีกว่า เพื่อที่จะได้เตรียมพร้อมกระโดดไปยัง 80586 ในวันข้างหน้า เมื่อเราก้าวมาถึง 80486 คงจะต้องทำความรู้จักหน้าตาความสามารถของ 80486 กันเสียก่อนและต้องรู้ไปถึงว่าเจ้า 80486 ตัวนี้มีอะไรแตกต่างกับ 80386 ตัวเดิมที่เรารู้จักมาแล้วบ้าง มาเริ่มต้นกันเลยนะครับ

80486 สร้างขึ้นมาด้วยเทคโนโลยี CHMOS IV ขนาด 1 ไมครอน มีทรานซิสเตอร์ 1.2 ล้านตัว มีคุณสมบัติดังนี้

1. มีแคชอยู่บนตัว 80486 เอง ดังรูปที่ 1

- ขนาดของแคช คือ 8 กิโลไบต์ ใช้ลักษณะการเข้าถึงแบบ four-way และมีขนาดไลน์ 16 ไบต์ (คำว่า four-way และไลน์ (line) จะกล่าวภายหลัง
- การทำแคชนี้เป็นการทำทั้งคำสั่ง (code) และข้อมูล (data)
- มีบัฟเฟอร์ 4 ตัวเพื่อทำ write-through update policy (จะกล่าวถึงคำนี้ภายหลังเช่นกัน)
- มีขา (pin) ที่จะทำ external flush, disabling และ line invalidation (snooping)

2. มีตัวประมวลทางคณิตศาสตร์บนตัว 80486 เอง (Floating Point Unit:FPU) ดังรูปที่ 2

- มีความสามารถตาม IEEE 754
- ซอฟต์แวร์เข้ากันได้ (compatible) กับ 387DX ร้อยเปอร์เซ็นต์

3. ทำ Burst-mode bus ได้ถึง 106 เมกะไบต์ต่อวินาที ในซีพียูขนาด 33 MHz

4. สนับสนุนการประมวลผลแบบมัลติโปรเซสเซอร์

- มีแคชในตัว (ทำให้การติดต่อกับอุปกรณ์หรือตัวประมวลผลอื่นได้ดียิ่งขึ้น)
- มีคำสั่งสนับสนุน (new instructions)
- มีขาสนับสนุน (new pins)
- สามารถต่อแคชคอนโทรลเลอร์เพิ่มได้ (second level cache controller)

5. ความสามารถ ดังรูปที่ 3

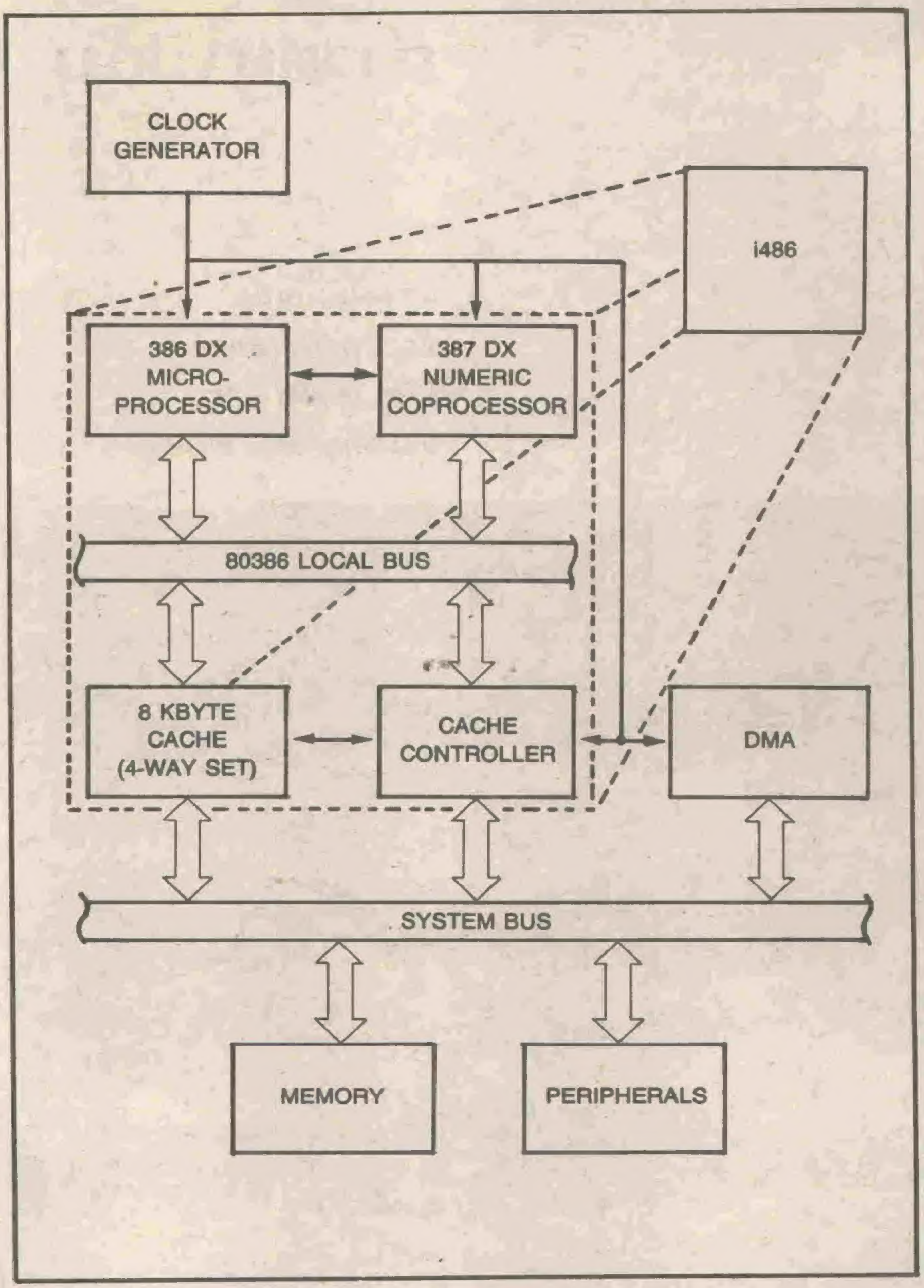
- เร็วกว่า 386DX ประมาณ 2-4 เท่า
- ประมวลผลได้ 15-20 VAX MIPS ที่ 25-33 MHz
- คำนวณได้ 1.6-2.0 MFLOP FPU

การทำงานในโหมดต่าง ๆ

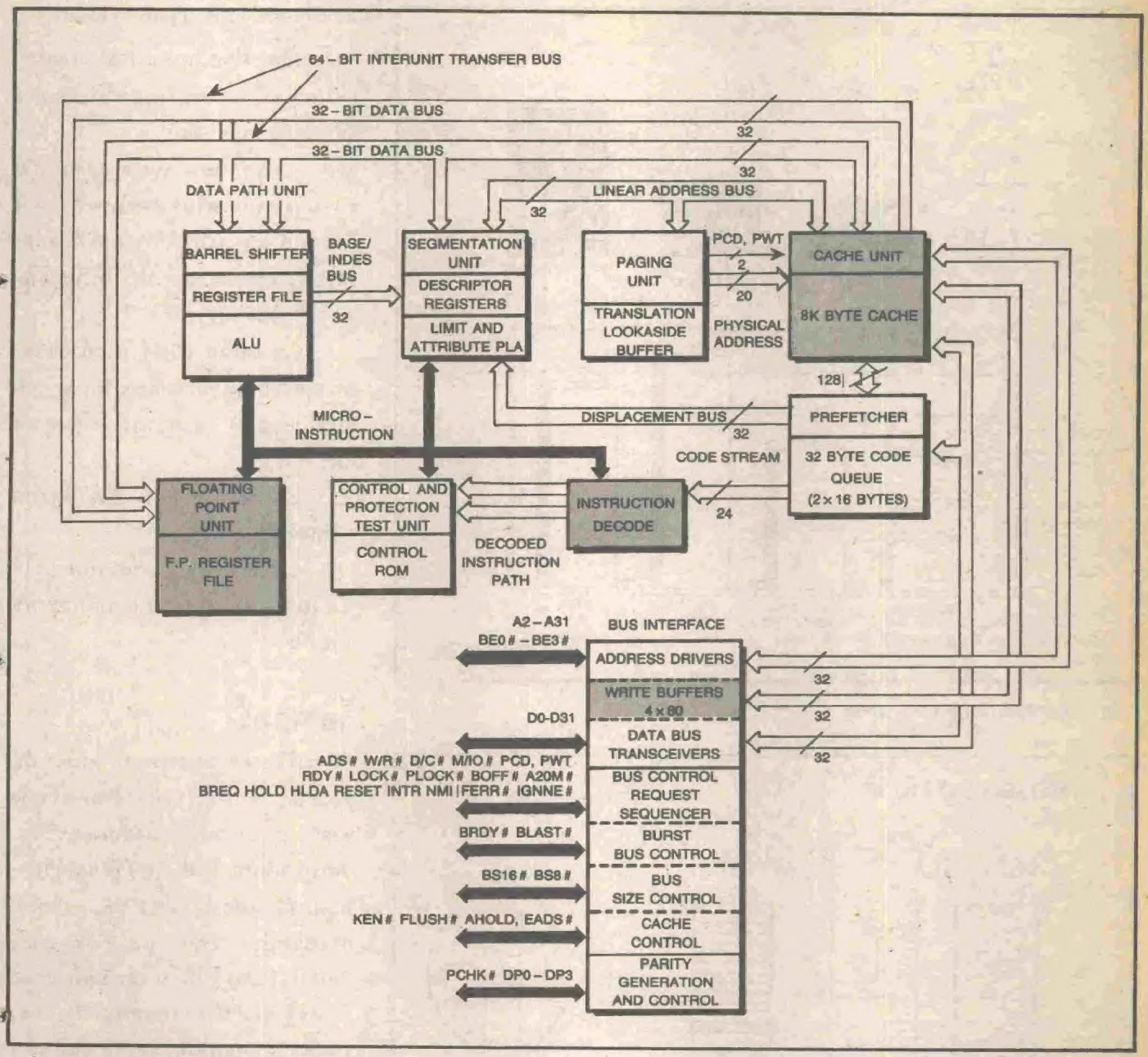
การทำงานของ 80486 มีโหมดการทำงานเช่นเดียวกับ 80386 คือ เรียลโหมด โปรเทคโหมด และเวอร์ชวล 8086 โหมด (จะถือว่ารวมอยู่ในโปรเทคโหมดก็ได้) ดังรูปที่ 4 ซึ่งความสามารถก็มีดังนี้

1. เรียลโหมด (Real mode)

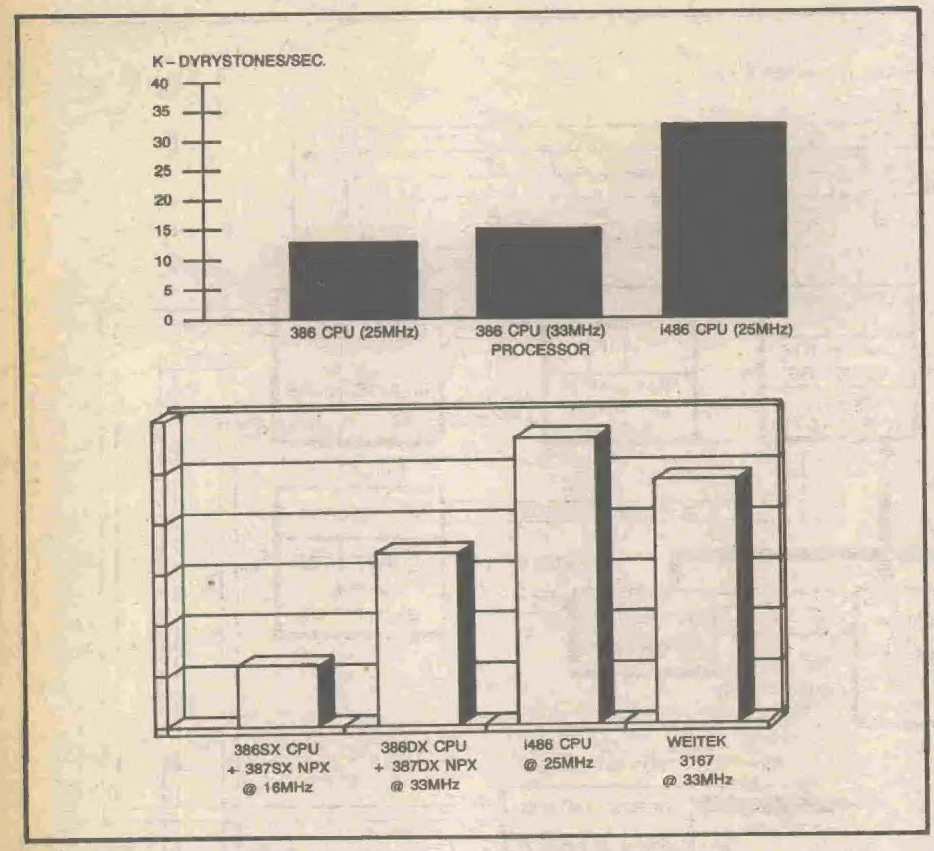
มีการทำงานเป็น 8086 ด้วยความเร็วสูง มีขนาดเซกเมนต์ 64 กิโลไบต์ มีการกระทำและการอ้างแอดเดรสเป็น 16 บิต และช่วงการอ้างแอดเดรสสูงสุด 1 เมกะไบต์เช่นเดียวกับ 8086



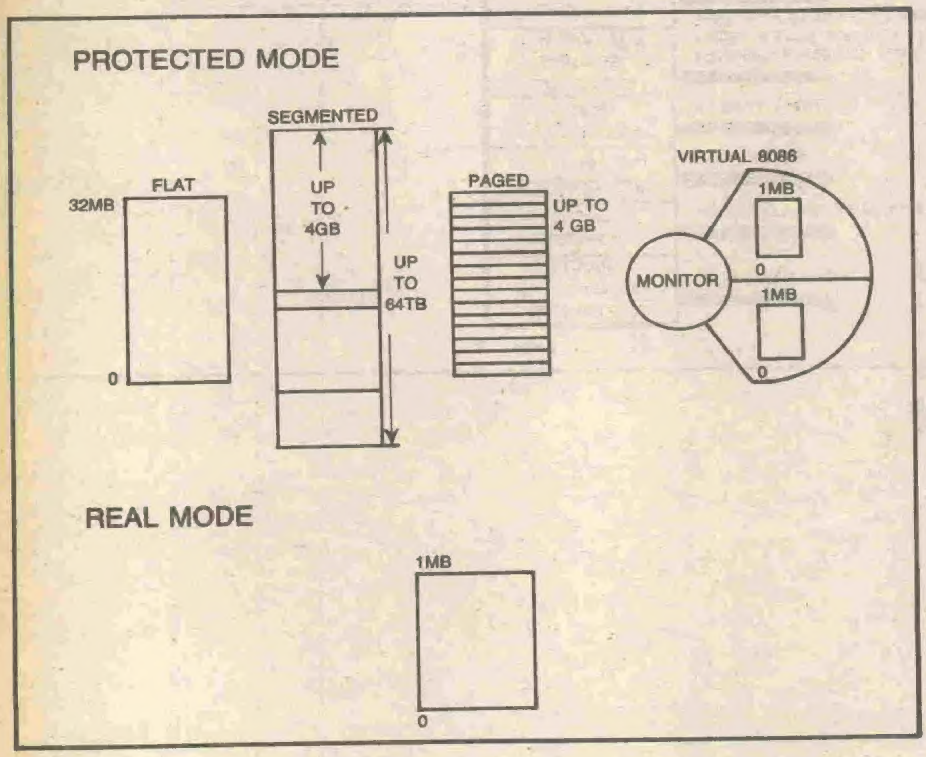
รูปที่ 1 บล็อกไดอะแกรมของ 80486



รูปที่ 2 บล็อกไดอะแกรมแสดงโครงสร้างภายในของ 80486



รูปที่ 3 กราฟแสดงความสามารถของ 80486



รูปที่ 4 โหมดการทำงานของ 80486

2. โปรเทคโหมด (Protected Mode)

เป็นการใช้ความสามารถของ 80486 เต็มที่ ซึ่งแล้วแต่การกำหนดลักษณะการใช้หน่วยความจำดังนี้

2.1 เซกเมนต์ (segmented) สามารถมีเซกเมนต์ได้ขนาดใหญ่สูงสุด 4 กิกะไบต์ และสามารถอ้างลอจิคัลแอดเดรส (เวอร์ชอลแอดเดรส) ได้ใหญ่ที่สุด 64 เทร่าไบต์ ดังรูปที่ 5

2.2 แฟลต (flat) การอ้างแอดเดรสแบบต่อเนื่อง (single linear address space) สามารถทำได้สูงสุด 4 กิกะไบต์

2.3 เพจ (paged) อ้างได้สูงสุด 4 กิกะไบต์

2.4 ไฮบริด (hybrid) เป็นการใช้หน่วยความจำแบบเซกเมนต์และเพจร่วมกัน

รีจิสเตอร์

รีจิสเตอร์ของ 80486 เหมือนกับ 80386 ยกเส้นการนำรีจิสเตอร์ของ 80387 เข้ามาร่วมอยู่บนตัวเองและรีจิสเตอร์เพิ่มขึ้นมาใหม่ คือ รีจิสเตอร์สำหรับการทดสอบ (TR3-TR5) และมีการปรับปรุง Translate look aside Buffer (TLB) ให้ดีขึ้น เช่นไม่ต้องหยุดทำเพจจิ้งขณะทำการทดสอบดังรูปที่ 6, 7 และ 8 นอกจากรีจิสเตอร์ที่เพิ่มขึ้นมาแล้ว รีจิสเตอร์บางตัวก็ได้ถูกแก้ไขเพิ่มเติมด้วยดังนี้

รีจิสเตอร์ CR0

เพิ่มบิต

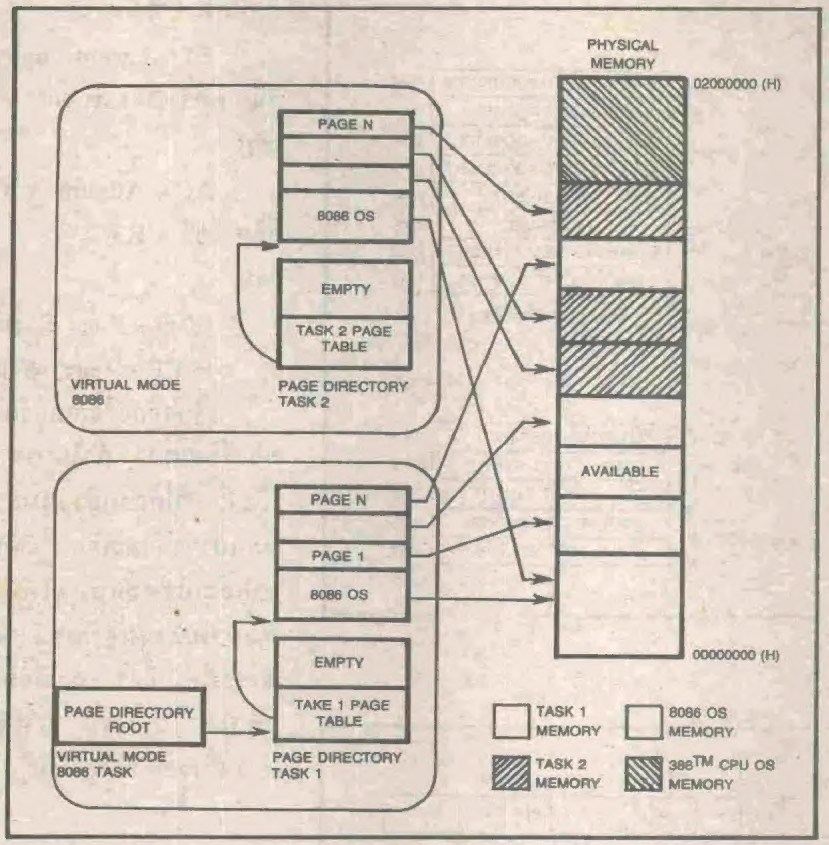
CD - Cache Disable

NW - No Write Through

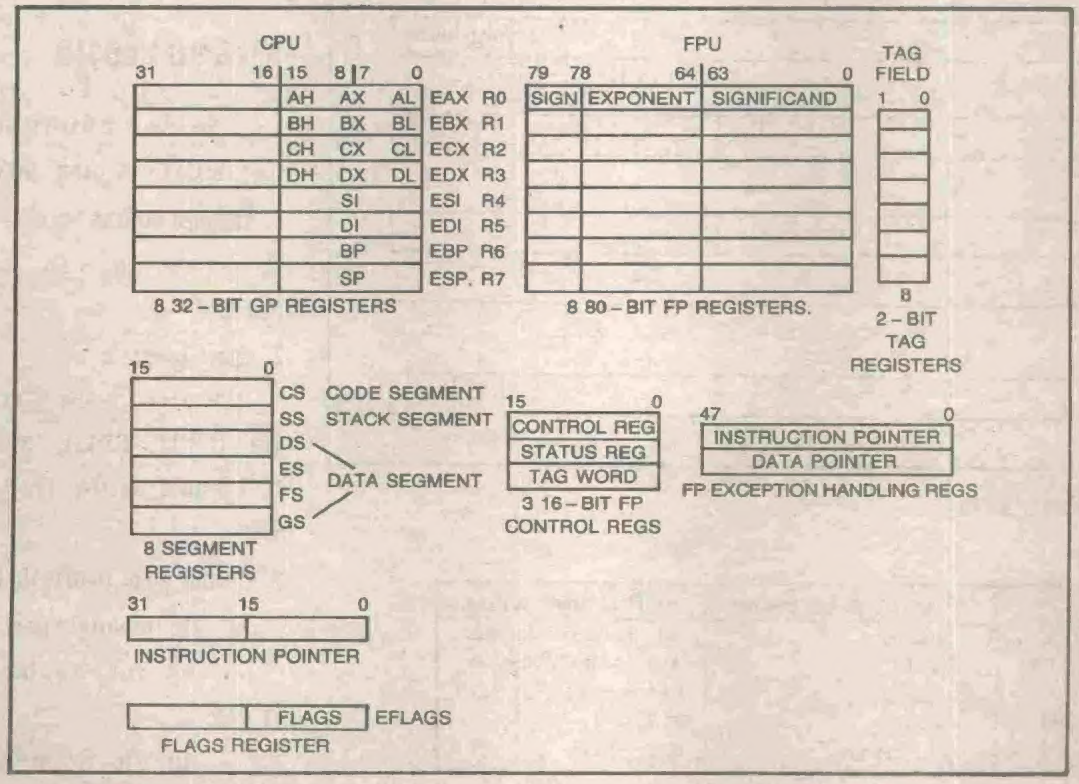
AM - Alignment Mask

WP - Write Protect

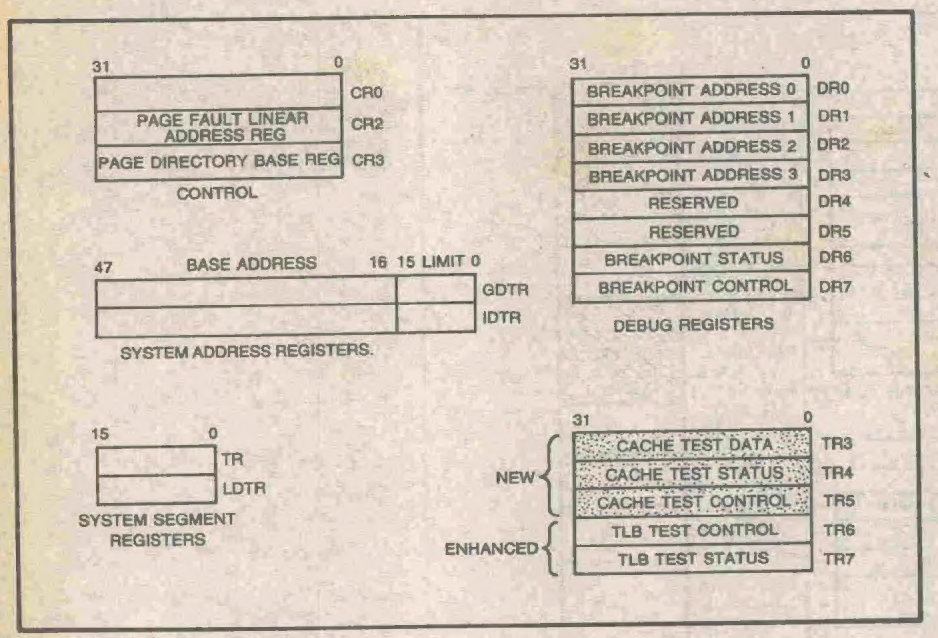
NE - Numerics Exception



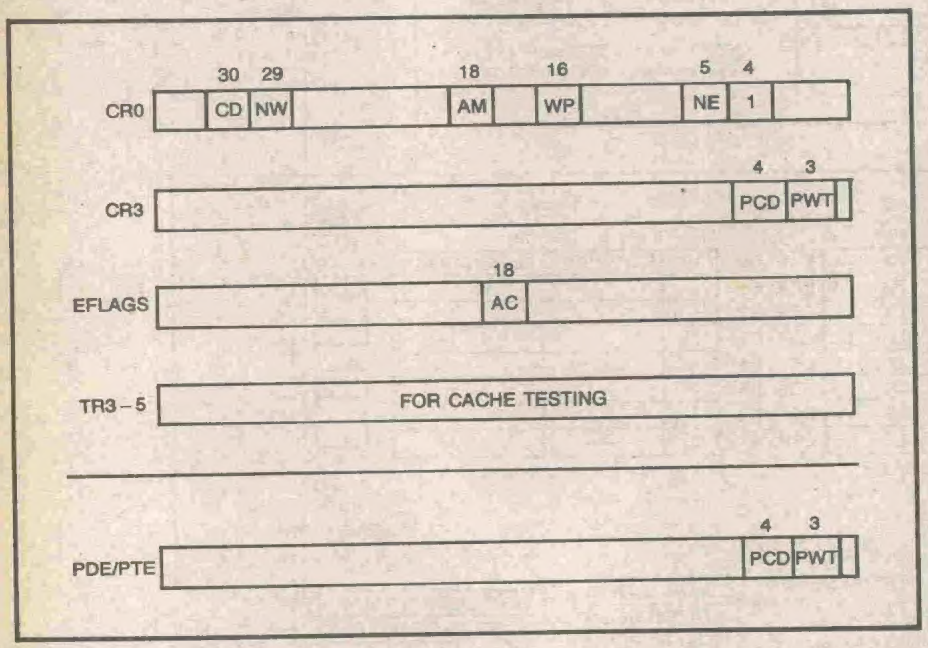
รูปที่ 5
ลักษณะการใช้หน่วยความจำแบบเวอร์ชวลโหมด



รูปที่ 6 รีจิสเตอร์ใช้งาน (Application level register)



รูปที่ 7 รีจิสเตอร์ระบบ (System level register)



รูปที่ 8 รีจิสเตอร์ที่ต่างจาก 80386

	16-Bit Addressing	32-Bit Addressing
BASE REGISTER	BX, BP	Any 32-bit GP Register
INDEX REGISTER	SI, DI	Any 32-bit GP Register Except ESP
SCALE FACTOR	none	1, 2, 4, 8
DISPLACEMENT	0, 8, 16 bits	0, 8, 32 bits

ตารางที่ 1 BASE และ INDEX รีจิสเตอร์ สำหรับการอ้างแอดเดรสแบบ 16 และ 32 บิต

ตัดบิตที่ 4 ทิ้งคือ

ET - Extention Type

รีจิสเตอร์ EFLAGS

เพิ่ม

AC - Alignment Check

รีจิสเตอร์ CR3

เพิ่ม

PCD - Page Cache Disable

PWT - Page Write Through

เรื่องของรีจิสเตอร์และการใช้งานของรีจิสเตอร์ ดังในตารางที่ 1, 2, 3 เป็นเรื่องที่ต้องกล่าวกันยาวมาก จึงไม่ขอกล่าวรายละเอียดในที่นี้ จึงขอยกรายละเอียดทั้งหมดให้ท่านผู้สนใจได้ศึกษาเองจากคู่มือหรือจากบทความซึ่งจะกล่าวถึงรายละเอียดของรีจิสเตอร์แต่ละตัวไป เช่น การจัดการหน่วยความจำของ 80386 ในไมโครฉบับที่ 51

ชนิดของข้อมูล

80486 สนับสนุนการใช้ข้อมูลของ 80386DX และ 80387DX ดังนี้

1. unsigned ordinal

- ขนาด 8, 16 และ 32 บิต (CPU)

2. signed integer

- ขนาด 8 บิต (CPU), 16 บิต (CPU, FPU), 32 บิต (CPU, FPU) และ 64 บิต (FPU) ดังตารางที่ 4

3. floating point (มาตรฐาน IEEE 754)

- 32 bit single precision (FPU)
- 64 bit double precision (FPU)
- 80 bit extended precision (FPU)

Register	Initial Value (BIST)	Initial Value (No Bist)
EAX	Zero (Pass)	Undefined
ECX	Undefined	Undefined
EDX	0400+Revision ID	0400+Revision ID
EBX	Undefined	Undefined
ESP	Undefined	Undefined
EBP	Undefined	Undefined
ESI	Undefined	Undefined
EDI	Undefined	Undefined
EFLAGS	00000002h	00000002h
EIP	0FFF0h	0FFF0h
ES	0000h	0000h
CS	F000h*	F000h*
SS	0000h	0000h
DS	0000h	0000h
FS	0000h	0000h
GS	0000h	0000h
IDTR	Base = 0, Limit = 3FFh	Base = 0, Limit = 3FFh
CR0	60000000h	60000000h
DR7	00000000h	00000000h
CW	037Fh	Unchanged
SW	0000h	Unchanged
TW	FFFFh	Unchanged
FIP	0000000h	Unchanged
FEA	0000000h	Unchanged
FCS	0000h	Unchanged
FDS	0000h	Unchanged
FOP	000h	Unchanged
FSTACK	Undefined	Unchanged

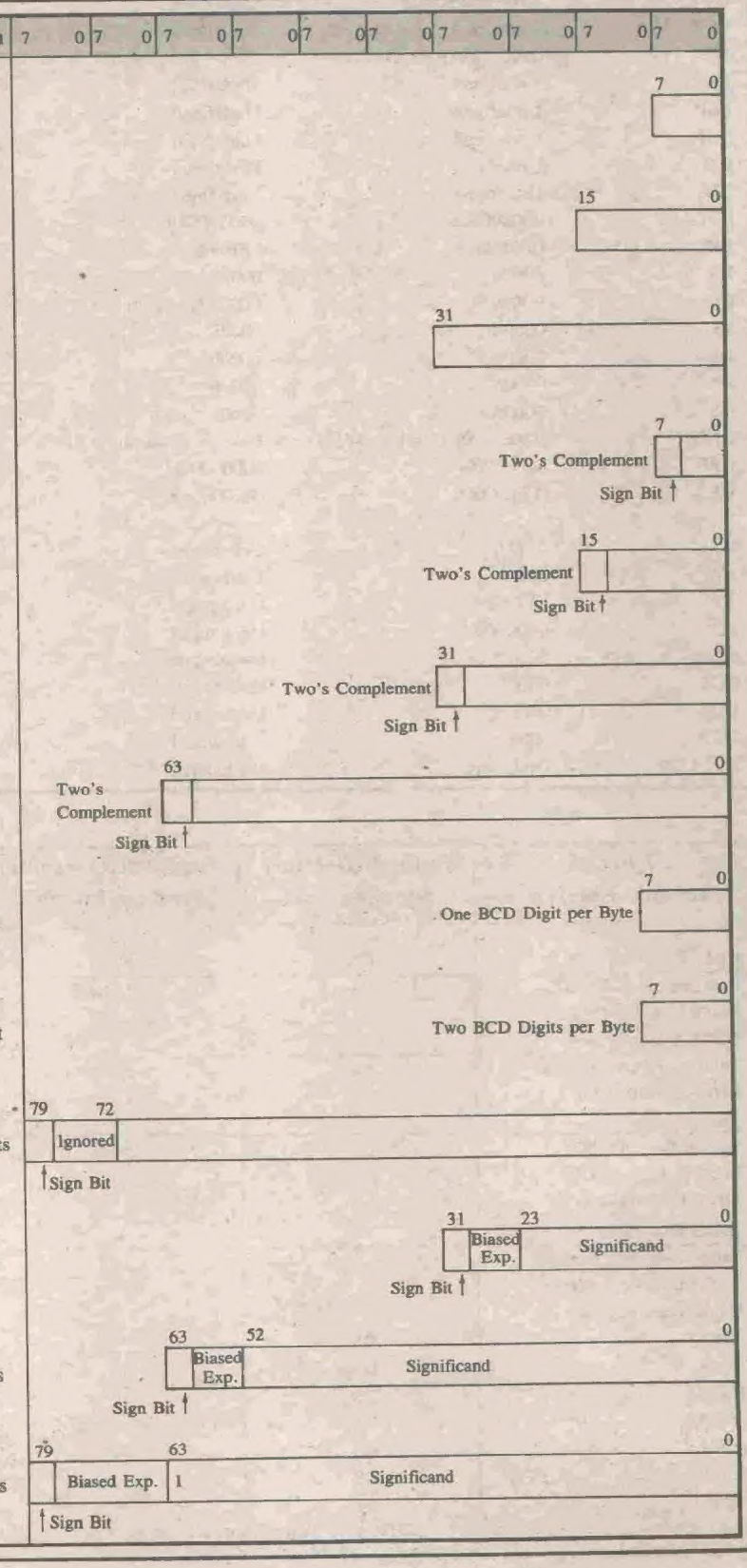
ตารางที่ 2 ค่ารีจิสเตอร์หลังการ reset

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS Instructions (DI is Base Register)	ES	None
Other Data References, with Effective Address Using Base Register of:		
[EAX]	DS	All
[EBX]	DS	
[ECX]	DS	
[EDX]	DS	
[ESI]	DS	
[EDI]	DS	
[EBP]	SS	
[ESP]	SS	

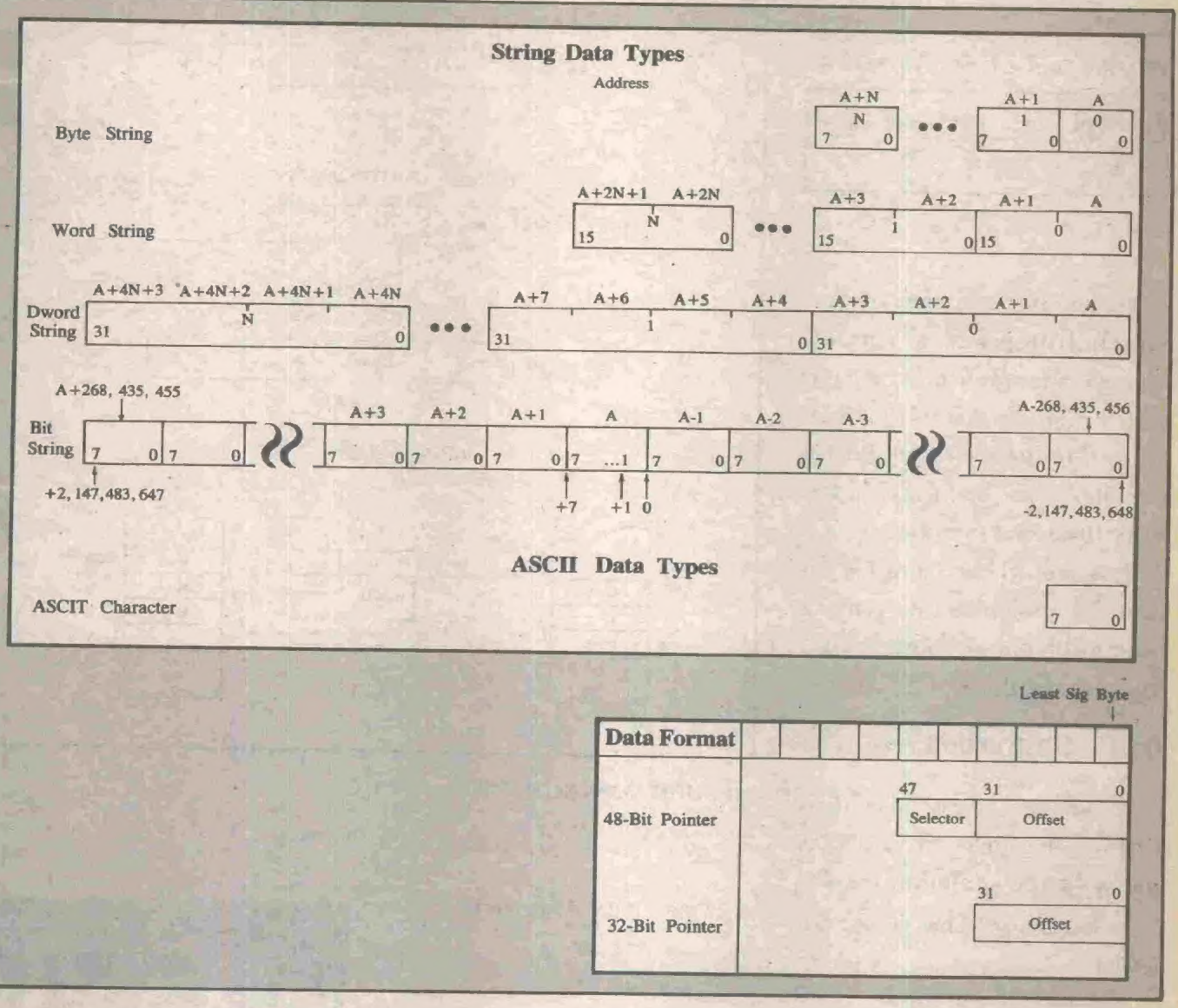
ตารางที่ 3 กฎการเลือกเซ็กเมนต์รีจิสเตอร์

ตารางที่ 4 ชนิดของข้อมูลที่ 80486 สนับสนุน

Data Format			Range	Precision
Byte	X		0-255	8 bits
Word	X		0-64K	16 bits
Dword	X		0-4G	32 bits
8-Bit Interger	X		10^2	8 bits
16-Bit Interger	X	X	10^4	16 bits
32-Bit Interger	X	X	10^9	32 bits
64-Bit Interger		X	10^{19}	64 bits
8-Bit Unpacked BCD	X		0-9	1 Digit
8-Bit Packed BCD	X		0-9	2 Digit
80-Bit Packed BCD		X	$\pm 10^{\pm 18}$	18 Digits
Single Precision Real		X	$\pm 10^{\pm 38}$	24 Bits
Double Precision Real		X	$\pm 10^{\pm 308}$	53 Bits
Extended Precision Real		X	$\pm 10^{\pm 4932}$	64 Bits



ตารางที่ 4 (ต่อ)



4. binary coded decimal
   - 8 bit packed/unpacked (CPU)
   - 80 bit packed (FPU)
5. string
   - Byte, WORD, DWORD (1-4 gigabyte)
   - bit (สูงสุด 4 gigabyte)
6. ASCII
   - byte (CPU)

Big Endian และ Little Endian

ซีพียูตระกูล 80×86 ที่ผ่านมาจะใช้ข้อมูลแบบ Little Endian นั่นคือ ไบต์ล่างจะอยู่ที่แอดเดรสล่าง ๆ และไบต์สูงจะอยู่ที่แอดเดรสสูง

เช่นเราเขียนโปรแกรมว่า

```
mov AX, 3231h
```

ในหน่วยความจำก็จะเป็นดังตารางที่ 5

ตำแหน่งที่	n	n+1
ข้อมูล	31	32

ตารางที่ 5

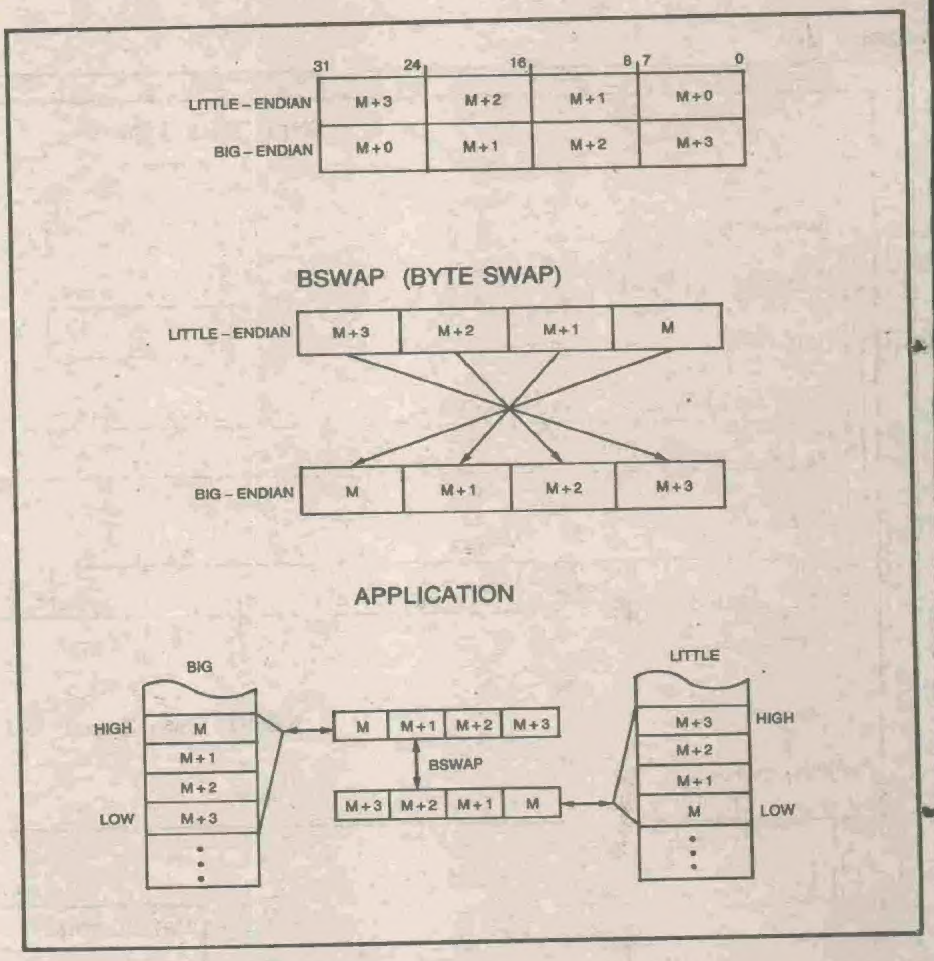
เช่นเดียวกันกับข้อมูลแบบ DWORD = '4321'

ในหน่วยความจำก็จะเป็นดังตารางที่ 6

ตำแหน่งที่	n	n+1	n+2	n+3
ข้อมูล	'1'	'2'	'3'	'4'

ตารางที่ 6

นั่นคือ เรียงไบต์ล่างที่หน่วยความจำล่างไปหาไบต์สูงที่หน่วยความจำตำแหน่งสูง ซึ่งตรงกันข้ามกับ Big Endian ที่เรียงในทางตรงกันข้าม สำหรับ 80486 นั้นสนับสนุนทั้ง Little Endian และ Big Endian โดยมีคำสั่งใหม่ BSWAP (Byte Swap) เพื่อทำงานร่วมกับตัวประมวลอื่นที่ใช้ข้อมูลแบบ Big Endian ได้ นั่นคือสนับสนุนการทำงานแบบมัลติโปรเซสเซอร์ให้สมบูรณ์มากขึ้นนั่นเอง ดังรูปที่ 9



รูปที่ 9 LITTLE ENDIAN และ BIG ENDIAN

คำสั่ง (Instruction set)

คำสั่งของ 80486 มีความเข้าใจกันได้ (compatible) กับ 80386DX และ 80387DX และได้เพิ่มคำสั่ง ดังนี้

INVD (invalid line in code/data cache)

WBINVD เหมือนกับ INVD ต่างกันที่การนำเสนอ (Write-back and Invalidate data cache)

INVLPG (invalidates a page mapped into the TLB)

XADD (Exchange and Add)

CMPXCHG (Compare and Exchange)

BSWAP ใช้ในการสลับข้อมูลระหว่าง Big-endian กับ Little endian

นอกจากจะมีคำสั่งเพิ่มขึ้นมาแล้ว ยังมีโอเปอแรนด์เพิ่มขึ้นมาอีกคือ TR3, TR4 และ TR5 ตัวอย่างเช่น

```
mov TR5, EAX
mov EBX, TR3
```

อินเตอร์รัพต์

กลไกการเกิดอินเตอร์รัพต์ของ 80486 นั้นเหมือนกับ 80386 ดังตารางที่ 7 ยกเว้นมีการเพิ่มชนิด (type) ที่ 17 เพื่อใช้ในการทำ alignment checking และอินเตอร์รัพต์ชนิดที่ 16 จะเกิดภายในเท่านั้น และชนิดที่ 9 จะไม่เกิด ทั้งนี้เพราะตัวประมวลผลคณิตศาสตร์ (FPU) ได้รวมเข้ามาไว้ในตัว 80486 เองแล้ว

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	YES	FAULT
Debug Exception	1	Any Instruction	YES	TRAP*
NMI Interrupt	2	INT 2 or NMI	NO	NMI
One Byte Interrupt	3	INT	NO	TRAP
Interrupt on Overflow	4	INTO	NO	TRAP
Array Bounds Check	5	BOUND	YES	FAULT
Invalid OP-Code	6	Any Illegal Instruction	YES	FAULT
Device Not Available	7	ESC, WAIT	YES	FAULT
Double Fault	8	Any Instruction That Can Generate an Exception		ABORT
Intel Reserved	9			
Invalid TSS	10	JMP, CALL, IRET, INT	YES	FAULT
Segment Not Present	11	Segment Register Instructions	YES	FAULT
Stack Fault	12	Stack References	YES	FAULT
General Protection Fault	13	Any Memory Reference	YES	FAULT
Page Fault	14	Any Memory Access or Code Fetch	YES	FAULT
Intel Reserved	15			
Floating Point Error	16	Floating Point, WAIT	YES	FAULT
Alignment Check Interrupt	17	Unaligned Memory Access	YES	FAULT
Intel Reserved	18-32			
Two Byte Interrupt	0-255	INT n	NO	TRAP

* Some debug exceptions may report both traps on the previous instruction, and faults on the next instruction.

ตารางที่ 7 ค่าอินเตอร์รัพต์ต่าง ๆ

i486	80386	Comments
Datapath unit	Execution unit	Faster
Segmentation unit	(same)	Faster
Paging unit	(same)	• New caching bits • Faster
Prefetcher	Code prefetch unit	32-byte queue
Instruction Decode	Instruction Decode Unit	• Two-stage decode • Flushed for jmp, call
Bus Interface	Bus Interface Unit	• No bus pipelining • 21 new pins • No 387 interface • Burst mode added • Four write buffers • Eight-bit bus support
Floating Point Unit	(none)	• Faster than 387 DX • 64-bit interunit bus
Cache Unit	(none)	128 bits in one clock for prefetch hit
Control and Protection Test Unit	(similar)	Controls both FPU and Datapath Unit

ตารางที่ 8 โครงสร้างภายใน 80486 เทียบกับ 80386

โครงสร้างภายใน

โครงสร้างภายในประกอบด้วยหน่วยต่าง ๆ คล้ายกับ 80386 ดังรูปที่ 10 และ 11 ซึ่งจะดูความคล้ายคลึงและแตกต่างได้จากตารางที่ 8 โครงสร้างภายในของ 80486 ได้ทำการปรับปรุงให้มีความสามารถเหมือนตัวประมวลแบบ RISC เช่น การ push, POP, load, store, add substract, การกระทำทางลอจิกและการ shift สามารถทำได้ภายใน 1 clock แต่อย่างไรก็ตาม 80486 ก็ไม่ใช่ RISC ทีเดียว โดยดูจากข้อแตกต่างได้ดังตารางที่ 9 และ 10

แคช (Cache)

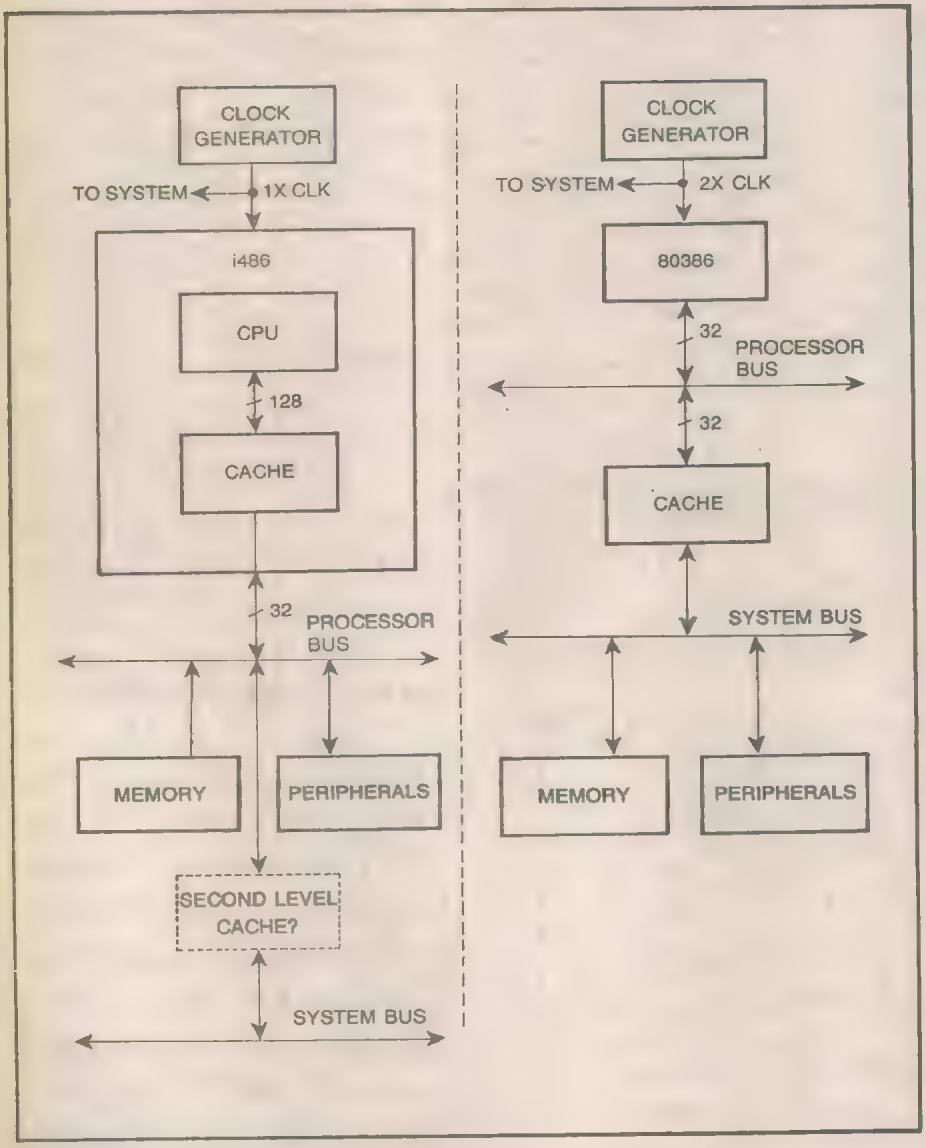
นอกจากจุดเด่นที่ 80486 รวมเอา 80387DX เข้าไว้ในตัวแล้ว จุดเด่นอีกอย่างก็คือ 80486 มีแคชอยู่ภายใน 8 กิโลไบต์ เราจึงควรศึกษาแคชไปพร้อมกันไปด้วย

จุดประสงค์และหลักการของแคช

จุดประสงค์ของแคชก็คือ การทำซีพียูมองเห็นหน่วยความจำที่ใช้อยู่ (ช้า) ให้เหมือนหน่วยความจำที่เร็ว คำว่าช้าก็คือ จะมี 1 หรือหลาย ๆ ช่วงเวลาหยุดรอ (wait states) และเร็วก็คือไม่มีช่วงเวลาหยุดรอ (zero wait state) หน่วยความจำที่เราใช้กันอยู่ก็คือ DRAM ซึ่งช้าแต่ก็ต้องใช้เพราะราคาถูกกว่าหน่วยความจำที่มีความรวดเร็วและราคาแพง นั้นคือ SRAM โครงสร้างของแคชจึงสร้างจาก SRAM การทำงานของแคชก็คือพยายามดึงข้อมูลที่ถูกใช้บ่อย (frequently) หรือพึ่งใช้ไปมาไว้บนตัวแคชเอง เมื่อซีพียูต้องการข้อมูลนั้น จึงสา-

RISC	i486™ Microprocessor
Load/store architecture	ALU operations allowed on memory operands
Fixed length instructions	One to fifteen bytes
Hardwired control	Has microcode
Small instruction set (~53)	141 integer instructions 73 FPU instructions
Minimalist hardware	BIST, TLB test register, on-chip cache test registers, breakpoint registers, ICE support

ตารางที่ 9 เปรียบเทียบระหว่าง 80486 กับ RISC



รูปที่ 10 โครงสร้างของระบบ 80486

มารถนำจากแคช (ซึ่งมีความเร็วสูง) ได้ทันที เช่นเดียวกับการเขียนข้อมูล แคชคอนโทรลเลอร์จะเป็นตัวจัดการให้ ดังรูปที่ 12 นอกจากจะรู้หลักการแล้ว เราควรรู้จักคำที่ใช้กับระบบแคชบ่อย ๆ ดังนี้คือ

line จำนวนหน่วยความจำที่น้อยที่สุดที่มีการเคลื่อนย้ายระหว่างแคชกับหน่วยความจำหลักในการทำปรับปรุง (update Cache)

miss ไลน์ที่อ้างถึงไม่ได้อยู่ในแคช

hit ไลน์ที่อ้างถึงอยู่ในแคช

การจัดโครงสร้างของแคช

แคช จะมีหน่วยความจำ 2 ประเภทอยู่บนตัวเองคือ

tag จะเป็นตัวบรรจุข้อมูลของ upper-order address ซึ่งเป็นตัวบอกหน้า (page) ของหน่วยความจำหลักที่จะใช้

date เป็นตัวเก็บข้อมูลที่เหมือนกับหน่วยความจำหลักที่ตำแหน่งตัวชี้ (index) ในหน้านั้น ขอให้ดูรูปไดเรค-แม็พประกอบ

นอกจากแท็กและดาต้าแล้วยังมีคำอีก 2 คำที่ใช้กับการจัดการแคช นั้นคือ

index เป็นตัวบรรจุ low order address ซึ่งเป็นตัวชี้ของข้อมูลในหน้าที่อ้างอิง

LRU (least-recently used) เป็นตัวบอกว่าไลน์ใดใน n-way cache ที่ถูกปรับปรุงหรือถูกใช้น้อยที่สุด ซึ่งเป็นตัวบอกว่าไลน์นั้นจะถูกเขียนทับเมื่อเกิด read miss

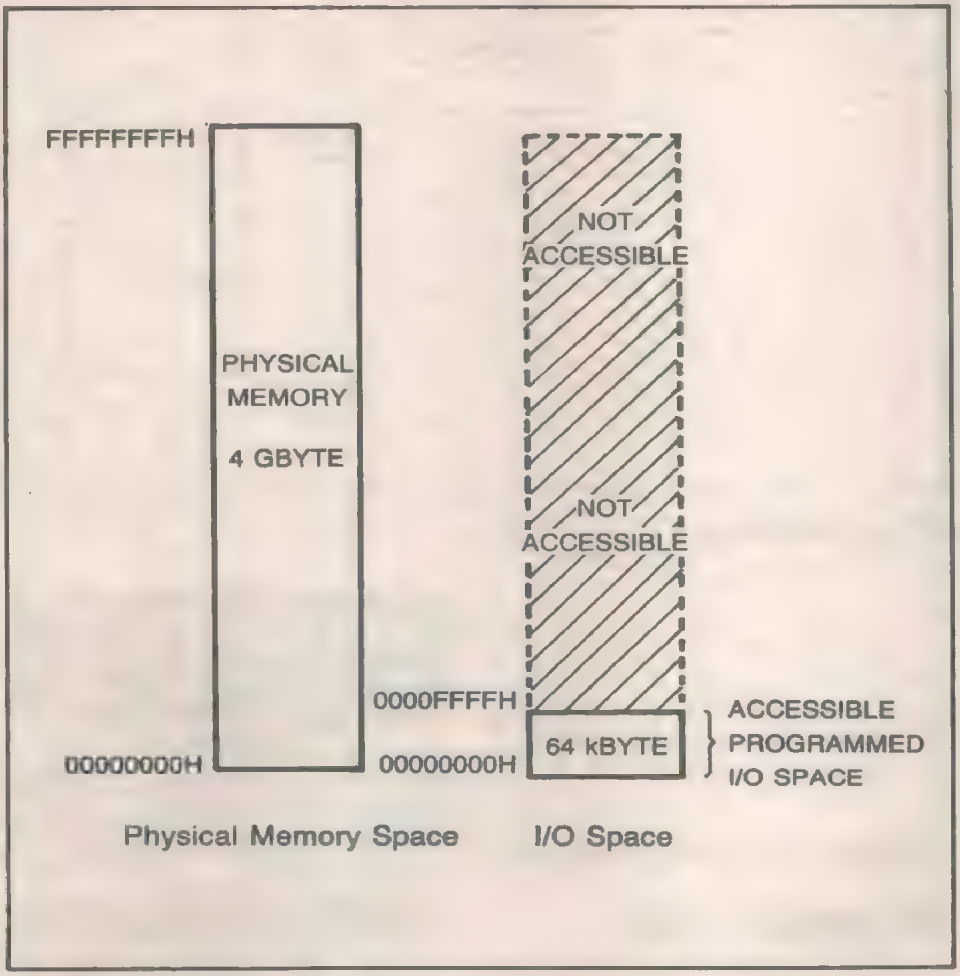
การจัดโครงสร้างของแคชจะเป็น

ไดเรคแม็พ (Direct Mapped)

คำอินเด็กซ์ที่จะให้ถูกจับคู่ (map-

Instruction Type	Clock Counts			
	386 DX CPU	i486 CPU	SPARC	88000 (est)
Load	4	1	2	1-3
Store	2	1	3	1
Reg/Reg	2	1	1	1
Jump	9/3	3/1	1/2	2/1
Call	9	3	3	1

ตารางที่ 10 จำนวน clock ที่ใช้ระหว่าง 80386, 80486 และ RISC บางตัว



รูปที่ 11 หน่วยความจำที่มีอยู่จริง (physical address) และ I/O

ped) บนหน้าเดียวเท่านั้นบนหน่วยความจำหลัก ซึ่งจะทำให้ไม่มี LRU เพราะถ้าไลน์ใด miss แล้วไลน์นั้นจะถูกเขียนทับ การ hit จะเกิดเมื่อค่า upper order address ที่ให้จากซีพียูตรงกับค่าเพจที่ตำแหน่งอินเด็กซ์ในหน่วยความจำแท็ก ดังรูปที่ 13

Two way set

ค่าอินเด็กซ์ที่ให้จะถูกจับคู่ได้ 2 กรณีในแต่ละทาง (way) ซึ่งเป็นการเพิ่มอัตราการ hit นั้นเอง (การ hit จะเกิดขึ้นเมื่อค่า upper order address ตรงกับค่าเพจในแท็กของ way ใด way หนึ่งในตำแหน่งอินเด็กซ์ที่ให้มา) LRU จะถูกปรับปรุงเป็นไลน์ที่เกิด miss ขึ้น

Four way set

เป็นการเพิ่มขยายของ two way set ทั้งนี้เพื่อเพิ่มอัตราการ hit นั้นเอง ใน 80486 ก็มี Four way set แคชอยู่ 2 ที่คือ Paging Unit (TLB) และ Cache Unit (code/data cache) ดังรูปที่ 14

Buffered WRITE และ SNOPPING

การจะทำให้หน่วยความจำหลักมีค่าสอดคล้องกับแคชขณะที่ซีพียูทำการเขียนข้อมูลกับแคชซึ่งเรียกว่า write-through update policy วิธีที่ง่าย ๆ ก็คือเขียนข้อมูลไปที่หน่วยความจำหลักทุกครั้งเมื่อเขียนไปที่แคช แต่ก็มีวิธีที่ดีกว่า โดยการเขียนไปที่ 'post' หรือ 'buffer' ของแคช ซึ่งข้อมูลที่เขียนไม่ใช่มีเพียงตัวข้อมูล (data) แต่รวมถึงแอดเดรสและค่าควบคุม (address and control information) ด้วย ซึ่งการเขียนไปที่บัฟเฟอร์นั้นแคชคอนโทรลเลอร์จะเป็นตัวจัดการเขียนไปที่หน่วยความจำหลักให้เอง ทำให้ซีพียูเขียนข้อ-

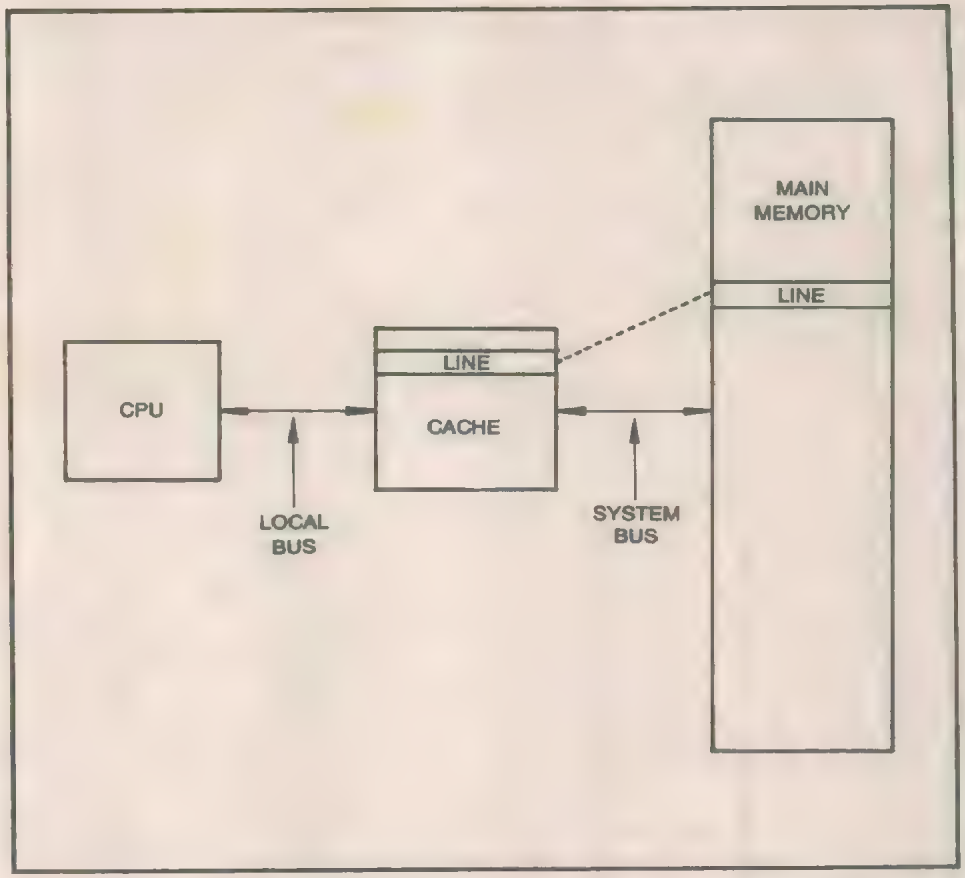
มูลด้วยความเร็วสูงที่ลอคัลบัส โดยแคชคอนโทรลเลอร์เขียนที่ซิสเต็มบัสให้เอง ดังรูปที่ 15

เมื่อระบบมีการใช้ตัวประมวลผล 2 ตัวหรือหลายตัวขึ้นไปกับระบบแคช จะทำให้เกิดปัญหากับการเขียนข้อมูลลงหน่วยความจำหลัก ถ้าข้อมูลที่เขียนเกิดจากตัวประมวลผลตัวอื่น (ตัวที่ไม่ได้มีแคช) บังเอิญตรงกับส่วนที่ทำการแคชไว้ จะทำให้หน่วยความจำหลักกับส่วนที่อยู่ในแคชไม่สอดคล้องกัน ซึ่งเราพอจะมีวิธีการแก้ได้ดังนี้

1. ทำให้หน่วยความจำที่จะใช้ร่วมกัน ไม่ให้เกิดการแคช (non cacheable) ทำได้โดยฮาร์ดแวร์ ซึ่งมีผลเสียกับความสามารถของระบบ

2. ใช้แคชร่วมกัน นั่นก็คือขณะที่ตัวประมวลตัวแรกใช้แคชอยู่ ตัวอื่นต้องหยุดรอก่อน ซึ่งมีผลให้เกิด wait state

3. Snoop โดยแคชคอนโทรลเลอร์จะคอยดูซิสเต็มส์บัส ในระหว่างการเขียนเมื่อตัวเองอยู่ในสภาวะ hold ถ้าตัวประมวลผลตัวอื่นเขียนข้อมูลที่แอดเดรสตรงกับในแคช ตัวควบคุมแคชจะทำการ invalidate line นั้น เพื่อให้เกิดปรับปรุงแคชเช่นเดียวกับ read miss update



รูปที่ 12 แนวความคิดของแคช

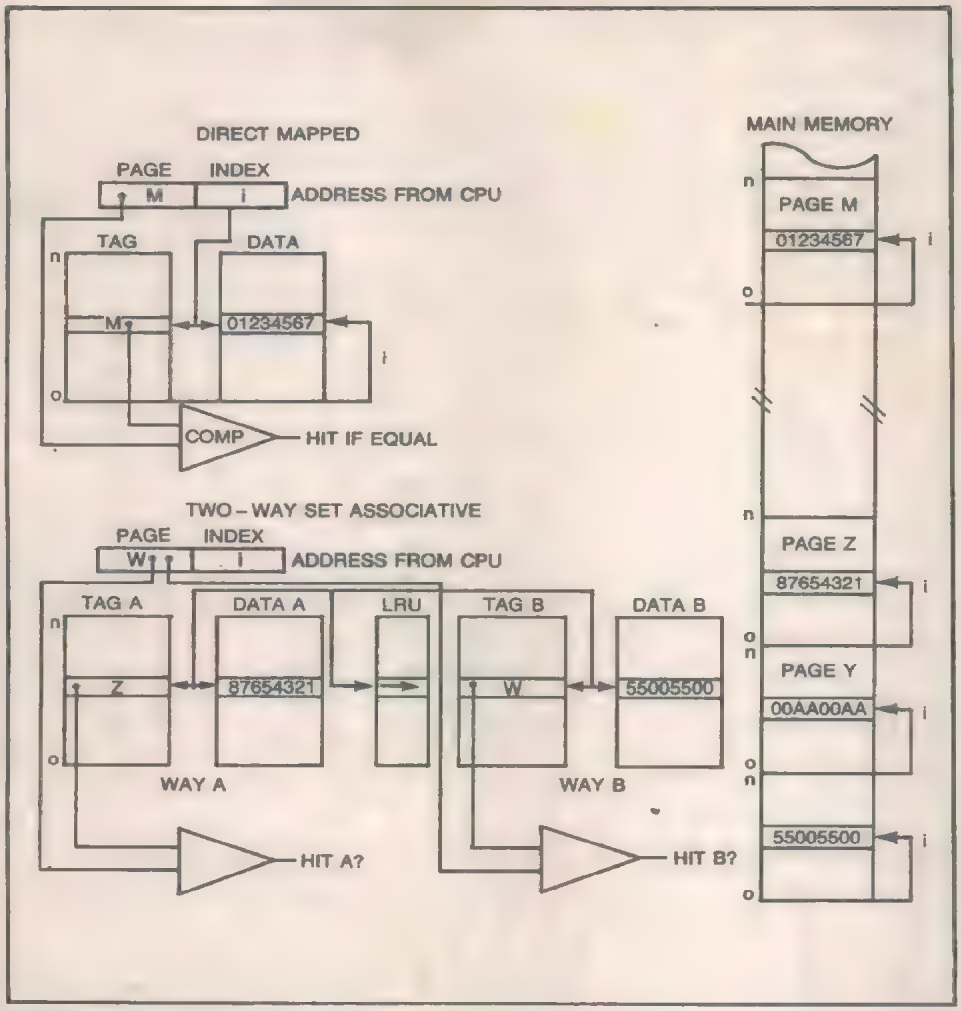
ขาสัญญาณของ 80486

80486 นั้นรวมเอา 80386 และ 80387 เข้าไว้ในตัวเองและแถมยังแคชในตัวอีก ทำให้ 80486 มีขาทั้งหมด 168 ขา ขณะที่ 80386 มีแค่ 132 ขา ดังรูปที่ 16 ซึ่งจะได้เห็นข้อแตกต่างได้จากตารางที่ 11 แต่กระนั้น 80486 จะไม่มีขาบางขาที่ 80386 มี ทั้งนี้เพราะไม่จำเป็นต้องมีนั้นเอง ดังตารางที่ 12

ข้อแตกต่างระหว่าง 80486 กับ 80386 + 80387

1. จำนวน clock ใน 1 คำสั่งจะถูกลดลง ทำให้มีประสิทธิภาพในการทำงานมากขึ้น ดังตารางที่ 13

2. ระบบบัสมีความเร็วขึ้นและมีข้อแตกต่างคือ 1X clock, parity support, burst cycles, cacheable cycles, cache invalidate cycles และ 8 bit bus support ดังรูปที่ 17



รูปที่ 13 directed mapped และ Two way set

3. มีแคชขนาด 8 กิโลไบต์อยู่ในตัว 80486 เอง ทำให้มีการเพิ่มบิต CD และ NW ในรีจิสเตอร์ CR0 และเพิ่มขาขึ้นเพื่อสนับสนุนแคช

4. มี 80387 อยู่ในตัว ทำให้ไม่ต้องทำ I/O เมื่อใช้คำสั่งเกี่ยวกับโฟลตติ้งพอยน์ อินเตอร์รัพต์ 9 ไม่สามารถเกิดขึ้นได้อีก ขณะที่อินเตอร์รัพต์ 13 เกิดขึ้นแทน

5. สนับสนุนการเกิดการผิดพลาดของโฟลตติ้งพอยน์เพิ่มขึ้น เพื่อความเข้ากันได้กับของเก่า (DOS Compatibility) ซึ่งทำให้ต้องการบิต NE ในรีจิสเตอร์ CR0 และขา (pin) เพิ่มขึ้นคือ FERR# และ IGNNE#

6. มีคำสั่งใหม่ 6 คำสั่งคือ

- **BSWAP** (Byte SWAP)
- **XADD** (Exchange and add)
- **CMPXCHG** (Compare and Exchange)
- **INVD** (Invalidate Data Cache)
- **WBINVD** (Write back and Invalidate data Cache)
- **IN__VLPG** (Invalidate TLB Entry)

7. รีจิสเตอร์ CR3 มีบิตถูกนิยามเพิ่มขึ้น 2 บิตคือ PCD และ PWT

8. มีเพจโปรเทคชันเพิ่มขึ้นซึ่งต้องการบิต WP เพิ่มขึ้นใน CR0

9. มี Alignment Check ทำให้เพิ่มบิต AC ใน EFLAG และบิต AM ในรีจิสเตอร์ CR0 รูปที่ 18 และ 19

10. เปลี่ยนวิธีการเกี่ยวกับ translation look aside buffer (TLB) โดยใช้วิธีการเช่นเดียวกับ on chip cache

11. มีรีจิสเตอร์ตรวจสอบเพิ่มขึ้นคือ TR3, TR4, และ TR5

12. prefetch queue เพิ่มจาก 16 ไบต์เป็น 32 ไบต์

13. หลังจาก reset, ID ใน upper byte ของรีจิสเตอร์ DX มีค่าเท่ากับ 04

ข้อมูลเกี่ยวกับ 80486 มีเรื่องที่น่าศึกษาอีกมากมายเช่นเดียวกับ 80386 ซึ่งคงจะนำมากล่าวในที่นี้ให้หมดคงเป็นไปไม่ได้ แต่ข้อมูลที่นำมากล่าวนี้ก็หวังเพียงเพื่อจะได้ก้าวต่อไปกับซีพียูเบอร์ใหม่ให้ทันกับเทคโนโลยีนั้นเอง หวังอย่างยิ่งว่าเราจะก้าวไปด้วยกันตลอดไปเพื่อเมืองไทยของเรา สวัสดีครับ

บ ท ความ พิเศษ

ก้าวต่อไปกับ 80486

Group	i486	386	Comments
CLK	CLK	CLK2	• TTL compatible • 1X clock
32-bit address bus	A2-A3 BE0#-BE3#	(same)	A4-A31 are bidirectional
Data bus	D0-D31	(same)	No byte-swap logic on-chip
Interrupt	Reset INTR, NMI	(same)	Different way to invoke self-test at reset
Bus arbitration	HOLD, HLDA BOFF#, BREQ#	HOLD, HLDA	New pins provide new fiexibility
Bus size control	BS8#, BS16#	BS16#	No byte swap logic on-chip
Bus cycle definition	M/IO#, D/C#, W/R# LOCK#, PLOCK#	M/IO#, D/C#, W/R#, LOCK#	• Different decode for M/IO #, D/C#, W/R# • PLOCK # imacates burst
Parity	PCHK# DP0-DP3	(none)	Even parity generation and check
Cache invalidation	AHOLD, EAD5#	(none)	Used in snooping
Cache control	KEN#, FLUSH#	(none)	• KEN# = cache enable • FLUSH # = 4 clock flush
Page caching control	PWT, PCD	(none)	Used with second-level cache
Numeric error reporting	FERR#, IGNNE #	(none)	DOS-compatibility pins for FPU errors
Address bit, 20 mask	A20M#	(none)	1MB wrap-around support for cache, etc.
Burst control	BRDY#, BLAST#	(none)	• Maintain burst • Signal end of burst

ตารางที่ 11 เปรียบเทียบขาของ 80486 และ 80386

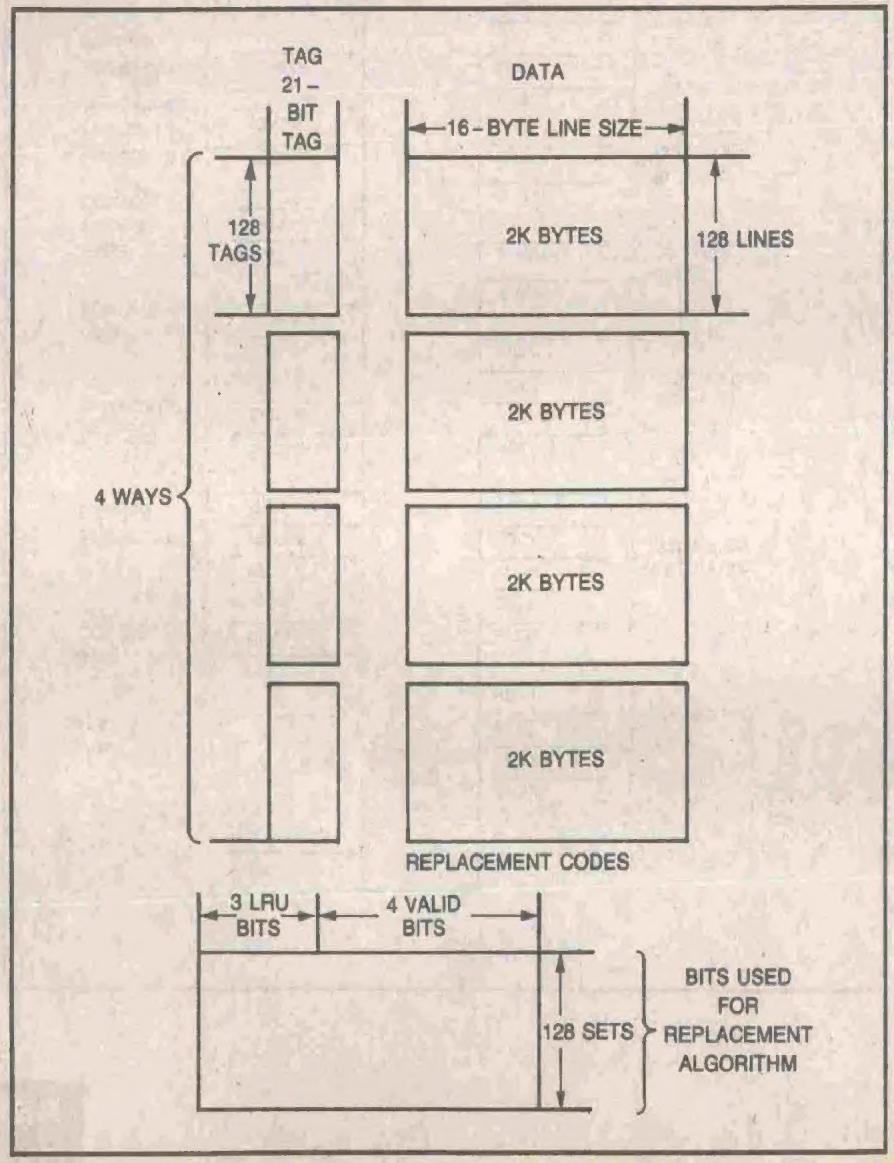
Pin Name	Implication
NA#	No pipelined cycles
PEREQ, ERROR#, BUSY#	No 387 support

ตารางที่ 12 ขาที่ 80386 มีแต่ 80486 ไม่มี

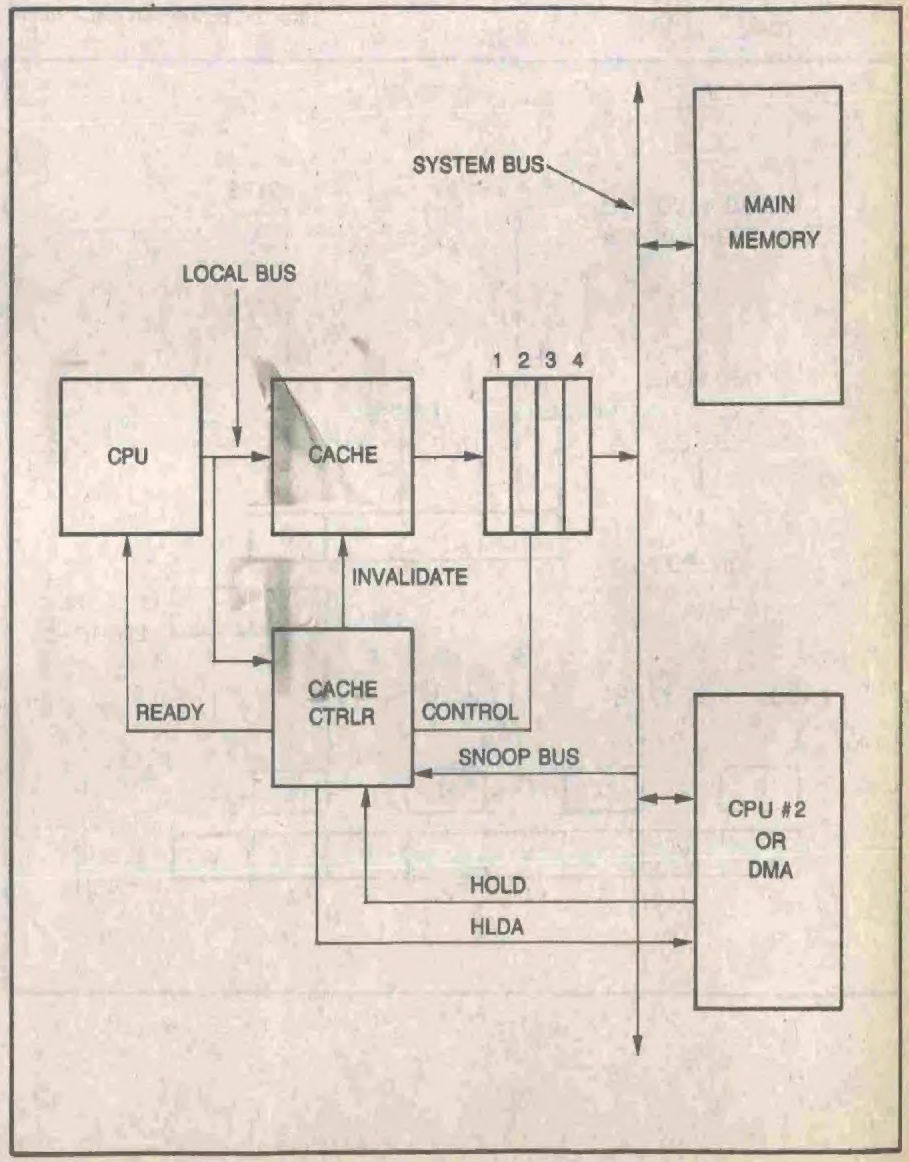
Instruction Type	Clock Counts	
	386™ CPU	i486™ CPU
LOAD	4	1
STORE	2	1
REG/REG	2	1
JUMP	9/3	3/1
CALL	9	3
Avg. CPI	4.5	1.8

Instruction Type	Clock Counts	
	387DX	i486™
FLD	14	3
FST	11	3
FADD/FSUB	23-31	10
FMUL	29-57	16
FDIV	88	73

ตารางที่ 13 จำนวน clock ที่ใช้ระหว่าง 80486 กับ 80386 + 80387

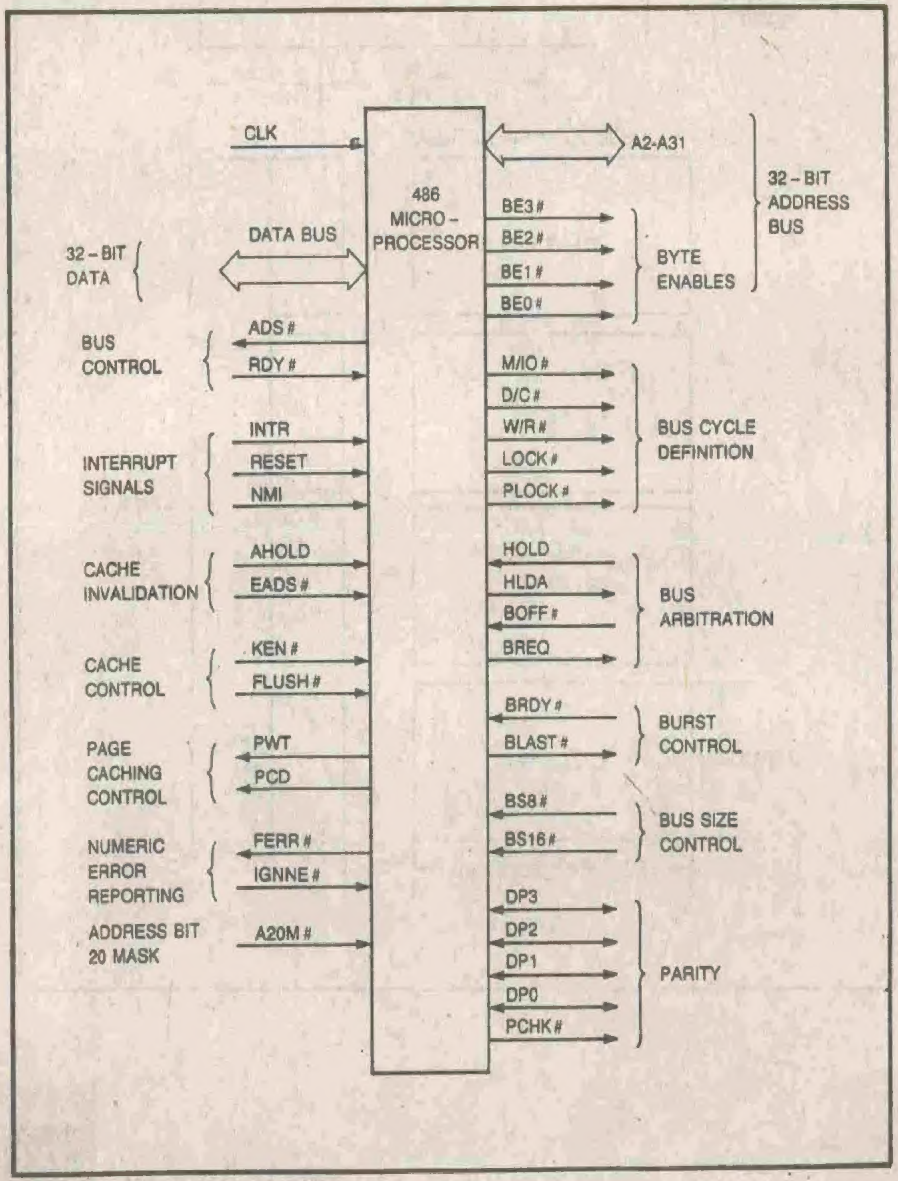


รูปที่ 14 four way set

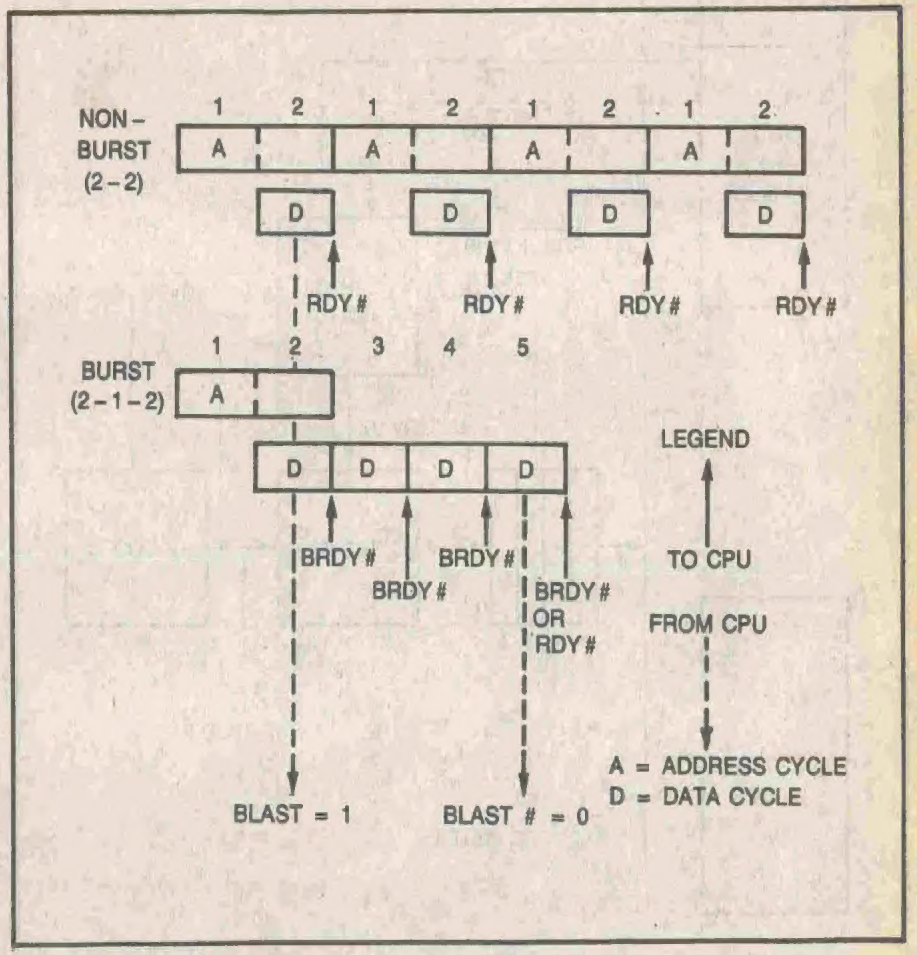


รูปที่ 15 buffer write และ snooping

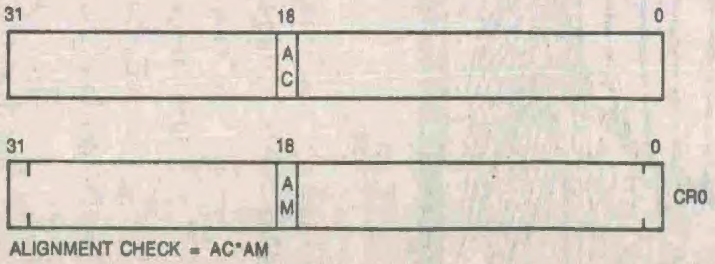
ก้าวต่อไปกับ 80486



รูปที่ 16 ขาต่าง ๆ ของ 80486



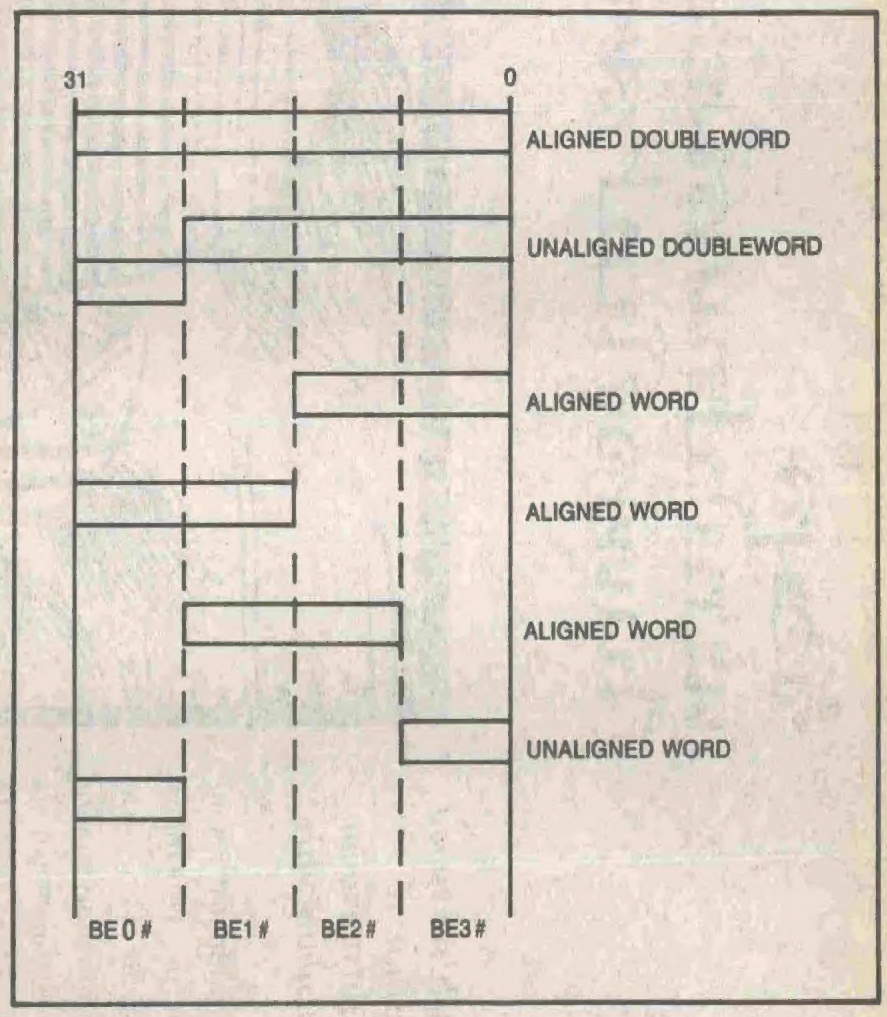
รูปที่ 17 BURST และ NON-BURST MODE



DATA TYPE ALIGNMENT REQUIREMENTS

Memory Access	Alignment (Byte Boundary)
Word	2
Dword	4
Single Precision Real	4
Double Precision Real	8
Extended Precision Real	8
Selector	2
48-Bit Segmented Pointer	4
32-Bit Flat Pointer	4
32-Bit Segmented Pointer	2
48-Bit "Pseudo-Descriptor"	4
FSTENV/FLDENV Save Area	4/2 (On Operand Size)
FSAVE/FRSTOR Save Area	4/2 (On Operand Size)
Bit String	4

รูปที่ 18 ค่า alignment



รูปที่ 19 การ alignment ของข้อมูล